81

POCKET PET

POCKET PET NOTIZIE

POCKET PET
anno 1 - numero 1
numero unico in attesa
di autorizzazione

Redazione:
Harden S.p.A.
Via Pirelli 11
Milano

SOMMARIO

Hanno collaborato a questo numero :

Alessandro de Simone
Bruno Brazzoduro
Giorgio Prada
Gloriano Rossi
Luciano Odoardi
Massimo Rossi
Roberto Brazzoduro
Roberto Sozzani

Direttore responsabile
Gloriano Rossi

Grafica, foto e stampa
Giorgio Prada

EDITORIALE

E noi stiamo li' a guardare

Ogni fiera, ogni manifestazione dedicata ai computer lascia un bagaglio di contatti, parole, spiegazioni. Ma a monte di tutto cio', quello che lascia veramente attoniti e' proprio il facile approccio che i ragazzini dimostrano verso il PET/CBM.
A -BIT 81-, e' chiaro, eravamo anche noi presenti ed il nostro stand e' stato letteralmente preso di assalto. Ma lo stupore maggiore non e' stato per quella quantita' di gente interessata che e venuta a trovarci, ma piuttosto, proprio in quella giornata di venerdi', per quelle scolaresche di scuola media e di scuola superiore.
10 anni, o poco piu', con il naso verso l'alto a guardarmi e con il ditino che punta al PET:
"Posso provare.. signore?"
"Bhe... si.. dai"
Non se lo fa ripetere due volte; tira fuori di tasca: un temperino, un pezzo di corda, quattro tappi di bottiglia, un pacchettino di figurine di Atlas UFO robot e poi... un foglietto di carta un po' stropicciata, la stende e cerca di stirarla con la mano destra tenedolo fermo con la sinistra.
Si sistema meglio sulla sedia e...

```
10 CLR : PRINT CHR$(147)
20 INPUT"......
30 ............
...............
275 GOSUB 1000
```

Si ferma un attimo: il listato, scritto a mano, e' finito; ma lui dopo una breve pensata, continua con la riga 280, poi la 290, e cosi' via.
Dopo un po' ... RUN

```
QUALE OPERAZIONE + - * / ?  e lui batte /
QUALE E' IL DIVIDENDO ?   batte 1342
QUALE E' IL DIVISORE  ?   batte 12
SYNTAX ERROR ON nnn
LIST nnn
```

"Accidenti !!!"
Corresse l'errore e poi di nuovo: RUN
Questa volta funziona.
Si rivolge ancora a me:
"Posso fare il listing sulla stampante?"
"Certo" rispondo io, e mi sono trattenuto dal chiedere "ma sei capace?", avrei senza dubbio fatto una figura del -put-.
Infatti: OPEN 4,4: CMD 4: LIST
"Grazie" , e se ne va tutto contento.
E noi? E noi siamo stati li' a guardare.

ASSEMBLER PER TUTTI

di Alessandro de Simone

Il sistema di numerazione usato nei μP e' quello binario puro e pertanto difficilissimo da rappresentare , ma con un piccolo artificio si puo' semplificare tutto notevolmente ricorrendo al sistema di numerazione esadecimale consistente di sedici simboli:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Per esempio il numero 12 in base dieci e' 1100 in binario puro e semplicemente C in esadecimale (fig. 1).

Poiche' (fig. 2) la memoria del computer e' una successione di gruppi di 8 bit ciascuno detti

parole o bytes

, noi dovremmo scrivere un programma in binario puro proprio come indicato in figura 2; figurarsi la confusione che ne verrebbe fuori per un programma di un centinaio di parole.
Se pero' dividiamo idealmente con una linea tratteggiata la 'striscia' otteniamo due gruppi di quattro bit ciascuno per ogni parola che , tradotti in codice esadecimale, sono molto piu' facili da scrivere in memoria e da controllare in seguito. Le prime tre parole di figura 2 anziche' 10101001 11111111 10001101 si scrivono A9 FF 8D ecc.. Naturalmente c'e' bisogno di un programma che traduca da codice esadecimale, scritto mediante la tastiera del PET utilizzando cifre e lettere, in binario puro dato che il computer ragiona solamente con questo sistema.

numero decim.	numero esadec.	binario puro
0	0	0 0 0 0
1	1	0 0 0 1
2	2	0 0 1 0
3	3	0 0 1 1
4	4	0 1 0 0
5	5	0 1 0 1
6	6	0 1 1 0
7	7	0 1 1 1
8	8	1 0 0 0
9	9	1 0 0 1
10	A	1 0 1 0
11	B	1 0 1 1
12	C	1 1 0 0
13	D	1 1 0 1
14	E	1 1 1 0
15	F	1 1 1 1

Figura 1

indir. base 10	ind.base 16 zona alta bassa	dato contenuto nell'indirizzo binario	esad.
826	03 3A	1010 1001	A9
827	03 3B	1111 1111	FF
828	03 3C	1000 1101	8D
829	03 3D	0000 0000	00
830	03 3E	1000 0000	80
831	03 3F	0110 0000	60
832	03 40	....	..
833	03 41	....	..
...	.. ..	....	..
...	.. ..	....	..

Figura 2

Tale programma , piuttosto sofisticato , e' residente nei PET dotati di nuove ROM e richiamabile battendo l'istruzione SYS(4); nei PET con vecchie ROM e' necessario invece caricare in memoria il programma noto con il nome di TIM.
Diamo comunque per scontato che il lettore sappia usare tale programma.

L'esempio considerato in figura 2 e' un semplicissimo programma in **L.M.** (linguaggio macchina). Prima di commentarlo definiamo l'accumulatore: questo e' un registro di otto bit interno alla CPU (6502) attraverso il quale viene elaborata una grande quantita' di informazioni presenti in un programma. Infatti quando facciamo partire il prg. A9 FF 8D 00 80 60 la CPU carica il dato A9, e poiche' le abbiamo detto che la locazione 033A contiene il primo dato da elaborare , capisce che A9 e' una istruzione ed esattamente, leggendo A9, carica nell'accumulatore la parola contenuta nella successiva locazione (033B) cioe' FF.

Tale concetto e' di fondamentale importanza: la CPU non puo' sapere se una certa locazione di memoria contiene un dato oppure una istruzione. Sa solamente (perche e' microprogrammata in tale modo) che la parola di partenza del programma deve considerarla coma una istruzione e continua l'elaborazione con questo presupposto.

Se infatti facciamo girare il programma da 033B anziche' da 033A, la CPU "pensa" che FF sia l'istruzione da eseguire e va subito in 'tilt' perche' non esiste questa istruzione fra le 151 istruzioni del 6502.

L'istruzione A9 e' una istruzione a due byte, cioe' quando la CPU legge A9, sa che la locazione di memoria successiva contiene un dato e non una istruzione; caricato, allora, il dato FF nell'accumulatore, leggera' il contenuto di 033C, che e' 8D, istruzione, questa volta, a tre byte.

Una istruzione a due byte viene conteggiata nel modo seguente: comando + dato; cosi' similmente 8D,come tutte le istruzioni a tre byte, e' costituita da : comando + dato + dato.

Incontrando 8D la 6502 eseguira' il comando trascrivendo il dato presente nell'accumulatore, nella locazione di memoria il cui indirizzo e' rappresentato, in esadecimale, dai dati contenuti nelle due locazioni di memoria successive a quella contenente il comando 8D.

In parole un po' piu' semplici la CPU scrive FF nella locazione il cui indirizzo e' 8000 esadecimale.
Attenzione : il primo byte successivo a 8D contiene la parte bassa dell'indirizzo (LA=low address) mentre la terza parola contiene la parte alta (HA=high address).

Dato che vogliamo scrivere il dato FF ,che e' in accumulatore, nella locazione 8000 scriviamo 8D 00 80 e non 8D 80 00. Tale modo di fare e' una caratteristica di tutte le istruzioni a tre byte che incontreremo: prima si scrive la LA poi la HA.

Dopo aver eseguito l'istruzione 8D la CPU incontra l'istruzione 60 (istr. a un solo byte) equivalente al RETURN. Poiche' non e' stata chiamata alcuna subroutine si ritorna automaticamente al BASIC uscendo in tal modo dal LM.
Su tale concetto torneremo piu' avanti; per ora basti sapere che se in un programma in LM c'e' un RETURN (60)... in piu', si ritorna al BASIC.

Vediamo ora a che cosa serve il programma appena esaminato. La memoria di schermo e' rappresentata da 1000 locazioni di memoria (25 righe per 40 colonne) a partire dall'indirizzo 80 00. Scrivendo allora FF in 80 00, apparira' il simbolo "▚" nella prima locazione della memoria di schermo (in alto a sinistra). Modificando opportunamente il contenuto di 033B e facendo rigirare il prg. in LM faremo apparire nell'angolo in alto a sinistra tutti i simboli grafici del PET. Si

potrbbe obiettare che il medesimo risultato si poteva ottenere semplicemente mediante il BASIC :

```
10 PRINT "▚"
```

Cio' e' vero, pero' c'e' da fare una prima considerazione: per far apparire in LM il simbolo ▚, abbiamo utilizzato solamente 6 byte; con il BASIC invece occuperemmo ben 10 byte senza contare che si e' dovuto utilizzare l'interprete del BASIC impiegando molto tempo. La sostanziale differenza fra i due metodi si nota quando si vorra' riempire tutto lo schermo con il carattere desiderato. Con il BASIC scriveremmo, per esempio:

```
10 A=1
20 B=1000
30 PRINT "[clr]"
40 FOR I=A TO B
50 PRINT "▚";
60 NEXT
70 END
```

Eseguendo il programma si puo' nettamente seguire il formarsi del 1000 simboli.
Proviamo ora a caricare il programma in LM che segue:

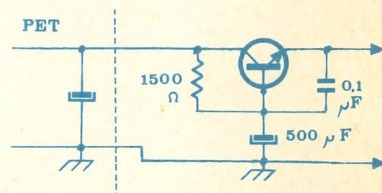
ind.part	dati
033A	A9 80 8D 48 03 A9 00 8D
0342	47 03 A9 FF 8D 00 80 EE
034A	47 03 D0 F6 AD 48 03 C9
0352	83 D0 01 60 EE 48 03 4C
035A	44 03 .. ..

```
100 I = 826
110 READ A$: IF A$="-1" THEN END
120 A1$ = LEFT$(A$,1)
130 A2$ = RIGHT$(A$,1)
140 IF A1$<"A" THEN 160
150 A1$ = STR$(ASC(A1$)-55)
160 IF A2$<"A" THEN 180
170 A2$ = STR$(ASC(A2$)-55)
180 A = VAL(A1$) * 16 + VAL(A2$)
200 POKE I,A: I = I+1
220 GOTO 110
230 DATA A9,80,8D,48,03,A9,00,8D
240 DATA 47,03,A9,FF,8D,00,80,EE
250 DATA 47,03,D0,F6,AD,48,03,C9
260 DATA 83,D0,01,60,EE,48,03,4C
270 DATA 44,03,-1
```

e.. SYS(826) ..... e' sorprendente vero ???!!!
..ma vediamo come gira :

Alimentazione prelevata direttamente dal PET-CBM

di R. Brazzoduro

Con questo brevissimo articolo vogliamo suggerire, per chi ha gia' una sia pur modesta conoscenza di hardware, il sistema per poter prelevare una tensione direttamente dal PET-CBM al fine di alimentare dei piccoli circuiti elettronici che sono applicabili direttamente al nostro computer.
La tensione di ingresso per il nostro curcuito si preleva dal grosso condensatore elettrolittico fissato sul telaio, (non saldato sul circuito stampato), che generalmente si aggira intorno ai 25000 µF di capacita'.
La tensione che e' qui presente e' di circa 8 Volt.

Il circuito che proponiamo dovra' essere applicato direttamente ai capi del condensatore. In pratica il piccolo condensatore di filtraggio che noi aggiungeremo viene moltiplicato per il guadagno del transistore (qualsiasi tipo NPN di potenza) che verra' usato.
Se il circuito alimentato da questa realizzazione fosse un amplificatorino, potrebbe accadere che dall'altoparlante escano dei disturbi; in questo caso sara' sufficente porre in parallelo all'alimentazione un condensatore da circa 100 nF, o valore similare, per eliminare completamente il disturbo.
Per concludere e' bene ricordare che la massa del telaio non e' la massa dei circuiti del PET-CBM, quindi per maggior sicurezza collegare la massa del circuito utilizzatore con la massa del telaio con un altro condensatore da 100 nF.

REMarks.

0333A A9 80 Nell'accumulatore viene scritto il numero 80 (in effetti viene scritto il numero binario puro 1000 0000 e non ripeteremo piu' questo concetto).

033C 8D 48 03 Il valore ora presente nell'accumulatore (80) viene riportato cosi' come e' nella locazione 03 48. Attenzione che nell'accumulatore e' ancora presente 80.

033F A9 00 Nell'acc. viene ora scritto 00 cancellando il precedente 80; naturalmente cio' che e' stato scritto in 03 48 **non** viene modificato.

0341 8D 47 03 Vedi istruzione 033C.

0344 A9 FF Vedi istruzione 033A oppure 033F.

0346 8D xx xx I due gruppi di xx stanno a significare che possiamo scrivere qualsiasi valore esadecimale; infatti quando in seguito faremo partire il programma, le istruzioni 033C e 0341 provvederanno a scrivere la parte alta e bassa dell'indirizzo; a questo punto dell'esecuzione del programma appare in alto a sinistra il simbolo "▚", che vogliamo trascrivere anche nella seconda posizione, terza ecc., fino a riempire tutto lo schermo. Dobbiamo pertanto fare in modo da impartire l'istruzione 8D 01 80 e poi 8D 02 80 .... fino a 8D FF 83. Per evitare di scrivere mille istruzioni (quante sono le celle dello schermo) utilizziamo un loop (fig. 3).

0349 EE 47 03 Questa istruzione (EE) aumenta di una unita' il contenuto della locazione il cui indirizzo e' 0347: da 00 passa a 01, oppure quando sara' contenuto A8 passera' ad A9 oppure (importante) da FF a 00.

034C D0 F6 L'istruzione D0 e' una istruzione a due byte che fa parte delle cosi' dette istruzioni di BRANCH (salto condizionato). Arrivati a questo punto, il proseguimento dell'elaborazione dipende dal risultato dell'ultima operazione effettuata. Nel caso specifico se EE 47 03 fornisce un valore uguale a 00 l'istruzione D0 viene ignorata ed il programma prosegue con l'istruzione successiva (034E). In caso contrario la CPU legge il primo bit del secondo byte dell'istru-

INDIR. ESA	OP. COD.	MNEMONICO
033A	A9	LDA# 80
033B	80	
033C	8D	STA 0348
033D	48	
033E	03	
033F	A9	LDA# 00
0340	00	
0341	8D	STA 0347
0342	47	
0343	03	
0344	A9	LDA# FF
0345	FF	
0346	8D	STA 8000
0347	00	
0348	80	
0349	EE	INC 0347
034A	47	
034B	03	
034C	D0	BNE
034D	F6	
034E	AD	LDA 0348
034F	48	
0350	03	
0351	C9	CMP# 83
0352	83	
0353	D0	BNE
0354	01	
0355	60	RTS
0356	EE	INC 0348
0357	48	
0358	03	
0359	4C	JMP 0344
035A	44	
035B	03	

Figura 3

zione (fig. 4 : 1111 0110) e se esso e' 0 eseguira' un salto in avanti mentre se e' 1 (nostro caso) fara' un salto indietro. Per salto si intende il numero di parole (e **non** di istruzioni) che bisogna escludere per rintracciare la successiva istruzione da eseguire. Nel nostro caso la CPU esegue un conteggio alla rovescia partendo dall'istruzione successiva a D0 (034E) definendola come FF. In seguito va all'indietro fino a che arriva al numero contenuto nel secondo byte di D0, cioe' F6, corrispondente all'ind. 0345. Il programma, allora, continuera' ad essere eseguito dalla **parola** successiva (sempre all'indietro) a 0345 e cioe' 0344 (A9). Bisogna pertanto prestare la massima attenzione al secondo byte delle istruzioni di BRANCH perche', in caso di errore, potremmo far continuare il programma da un dato anziche' da una istruzione oppure da una istruzione indesiderata. E' ovvio che nelle prossime puntate amplieremo questi concetti.
Notiamo facilmente che il blocco di istruzioni 0344-034D sara' eseguito 255 volte fino a che cioe' 0347 non conterra' 00 consentendo alla CPU di ignorare D0 F6. Ogni volta che il blocco viene eseguito lo schermo viene riempito di 256 simboli di "▚". Poiche' lo schermo e' di 1000 caratteri il blocco deve essere eseguito quasi quattro volte. Vediamo come:

034E AD 48 03 Carica in accumulatore il dato contenuto in 0348.

0344	A9		
0345	FF		F6 ↑
0346	8D		F7
0347	00		F8
0348	80		F9
0349	EE		FA
034A	47		FB
034B	03		FC
034C	D0		FD
034D	F6	1111 0110	FE
034E	AD		FF
034F	..	....	..
....	..	....	..
0353	D0		
0354	01	0000 0001	
0355	60		01
0356	EE		02 ↓
0357	..	....	..
....	..	....	..

Figura 4

0351 C9 83 Questa istruzione a due byte compara, mediante sottrazione, l'accumulatore con il numero esa 83; se i due numeri risultano uguali fornira' 00 come risultato (attenzione: l'accumulatore non viene comunque modificato ne' tantomeno il numero esa 83).

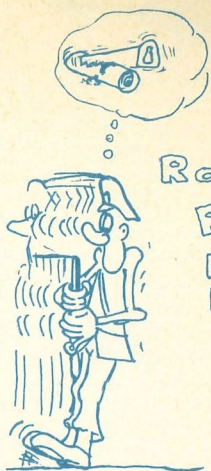
0353 D0 01 In questo caso, a differenza di 034C, il primo bit del secondo byte e' 0 (fig. 4) e pertanto nel caso in cui il risultato della comparazione 0351 sara' uguale a 00, sara' saltata una parola ed il programma continuera' da 0356.

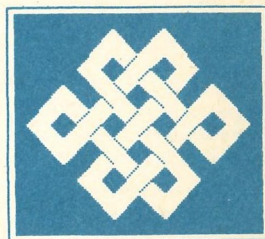
0355 60 Return subroutine: si esce dal L.M. e si ritorna al BASIC.

0356 EE 48 03 A questa istruzione (vedi 0349) si giunge se il risultato della comparazione 0351 indica che l'H.A. di 0346 (cioe' 8D 47 **80**) non e' ancora passato al valore 83, cioe' se siamo ancora nella memoria di schermo.
Riepilogando, il blocco 0344-034D viene eseguito 255 volte mentre 0344-0359 viene eseguito 4 volte prima di ritornare al BASIC.

```
0359  4C 44 03 istruzione   di   salto
                incondizionato:     la
                continuazione del pro-
                gramma passa   all'is-
                truzione  contenuta  in
                0344.
```

```
100 SYS(826)
110 GET A$: IF A$="" THEN 110
120 A= ASC(A$): POKE 837,A
130 GOTO 100
```

Il lettore potra' provare la validita' di questa routine in linguaggio macchina, in precedenza caricata, con il programmino qui sopra suggerito.

Simbolo della fortuna

Da una ricerca di Odoardi.

```
5 OPEN6,4,6:PRINT#6,CHR$(18):CLOSE6
10 OPEN4,4:CMD4
100 A1$="┌──────────────────┐"
110 A2$="|▣                  ▮|"
120 A3$="|▣       ◥▣ ◥▣       ▮|"
130 A4$="|▣      ◥ ▼  ◥▣     ▮|"
140 A5$="|▣     ◥ ◥◤╱ ◥◤ ◥▣    ▮|"
150 A6$="|▣      ▼ ▼╱▼ ◤     ▮|"
160 A7$="|▣   ◥▣ ▼ ╲◥◤ ╲◤ ◥▣   ▮|"
170 A8$="|▣  ◥  ▼╲ ▼╲ ▼  ◥▣ ▮|"
180 A9$="|▣ ◥ ◥◤╱ ◥◤╱ ◥◤╱ ◥◤ ◥▣ ▮|"
190 A0$="|▣  ▼ ▼╱▼ ▼╱▼ ◤ ▮|"
200 B1$="|▣   ◤ ◥◤ ╲◥◤ ╲◥◤  ◤  ▮|"
210 B2$="|▣    ▼ ◥╲ ▼╲ ◥▣ ▼  ▮|"
220 B3$="|▣     ◤ ◥◤╱ ◥◤ ◥▣    ▮|"
230 B4$="|▣      ▼ ▼╱▼ ◤     ▮|"
240 B5$="|▣       ◤ ◥◤  ◤       ▮|"
250 B6$="|▣        ▼  ▼       ▮|"
260 B7$="|▣                  ▮|"
270 B8$="└──────────────────┘"
300 PRINTA1$:PRINTA2$:PRINTA3$:PRINTA4$:PRINTA5$:PRINTA6$:PRINTA7$
310 PRINTA8$:PRINTA9$:PRINTA0$:PRINTB1$:PRINTB2$:PRINTB3$:PRINTB4$
320 PRINTB5$:PRINTB6$:PRINTB7$:PRINTB8$
330 PRINT#4:CLOSE4
400 OPEN6,4,6:PRINT#6,CHR$(24):CLOSE6
```

In Cina questo simbolo rappresenta la fortuna.
Il programmino proposto esegue, per chi ha la stampante 3022, la figura in oggetto.
Cancellando le righe 5, 10, 330 e 400 il simbolo apparira' sullo schermo.

ESPANSIONE
della data e dei numeri

PET & CBM 2001/3032

di Roberto Sozzani

Espansione della Data 1

```
100 REM ** ESPANSIONE DATA **
110 PRINT"[CLR]"
120 INPUT"[DOWN][DOWN][DOWN]DATA IN FORMA GGMMAA";DA$
130 FORI=1TO6:A$=MID$(DA$,I,1):IFA$<"0"ORA$>"9"THEN120
140 NEXT:MM=VAL(MID$(DA$,3,2)):IFMM>12ORMM=0THEN120
150 MM$=MID$("  GENFEBMARAPRMAGGIULUGAGOSETOTTNOVDIC",MM*3,3)
160 DA$=LEFT$(DA$,2)+" "+MM$+" 19"+MID$(DA$,5,2)
170 PRINT:PRINTDA$
180 GOTO120
```

Questa routine e' stata studiata per trasformare una data dalla forma numerica alla forma estesa:

da 010481 a 01 APR 1981.

Normalmente, infatti, per risparmiare spazio e tempo nell'introduzione dei dati nel computer, conviene esprimere una data utilizzando esclusivamente dei numeri che indichino il giorno, il mese e l'anno; e' preferibile, invece, sia per motivi estetici che di chiarezza dei tabulati, ottenere la stampa delle date in forma piu' chiara e completa.
Si rende quindi necessaria questa routine per effettuare le trasformazioni desiderate senza creare problemi all'operatore.
Al momento dell'inserimento in un programma, queste istruzioni potranno essere limitate alle sole righe 130,140,150 e 160.
Se un controllo di numericita' o comunque di validita' dell'informazione fosse gia' stato eseguito, allora anche il ciclo di FOR...NEXT contenuto nelle righe 130 e 140 potra' essere eliminato.
Da una breve analisi risulta evidente che le variabili chiave per l'espansione del mese siano MM e MM$: MM (riga 140) sara' uguale al valore delle due cifre che nella variabile introdotta indicano il mese (MID$(DA$,3,2)).
MM$, invece, sara' uguale a tre lettere opportunamente selezionate dalla lunga stringa di caratteri contenuta nella riga 150.
Facciamo un esempio:

Data	:	120681
MM	:	valore del numero formato dalla terza e quarta cifra, quindi MM = 06
MM$	:	sara' uguale a tre lettere estratte dalla stringa di caratteri che troviamo nella riga 150, a partire dal carattere nella diciottesima posizione (MM*3), che corrispondono a : GIU (non dimenticate di contare i due spazi iniziali !).

La riga 160 provvede ora a trasformare la variabile DA$ nella forma definitiva desiderata che sara':

12 GIU 1981

Espansione della Data 2

```
100 REM *********************
101 REM ** ESPANSIONE DATA **
102 REM *********************
103 :
110 DIMM$(12)
120 FORI=1TO12:READM$(I):NEXT
130 PRINT"♥"
140 INPUT"ⓆⓆⓆDATA IN FORMA GGMMAA";DA$
150 FORI=1TO6:A$=MID$(DA$,I,1):IFA$<"0"ORA$>"9"THEN140
160 NEXT:I=VAL(MID$(DA$,3,2)):IFI>12ORI=<0THEN140
170 DA$=LEFT$(DA$,2)+" "+M$(I)+" 19"+MID$(DA$,5,2)
180 PRINT:PRINTDA$
190 GOTO140
200 DATAGENNAIO,FEBBRAIO,MARZO,APRILE
205 DATAMAGGIO,GIUGNO,LUGLIO,AGOSTO
210 DATASETTEMBRE,OTTOBRE,NOVEMBRE,DICEMBRE
```

Questa seconda routine prevede l'espansione completa del mese, e non la sua abbreviazione.
I nomi dei mesi sono stati pertanto riportati come DATA nelle righe 200 e 210.
La variabile MM viene qui sostituita dalla variabile I, mentre M$(I) corrispondera' al mese prescelto.
Nella parte iniziale vengono date le istruzioni per caricare nell'opportuna tabella i valori alfanumerici dei relativi mesi (righe 110,120).
Valgono qui tutte le considerazioni fatte per la routine precedente.

Espansione dei numeri

A volte, purtroppo, e' necessario scrivere dei numeri in sottoforma di lettere e cio' crea spesso dei problemi poiche' ci troviamo di fronte a parole particolarmente lunghe. Chi non ha mai avuto esitazioni nel compilare un assegno scagli la prima pietra.
Nei centri meccanografici queste scritte vengono eseguite automaticamente su documenti, assegni ecc. utilizzando routines alquanto complesse.
Anche il nostro buon PET puo' fare tutto cio' e per di piu' con una routinetta che, al confronto di quelle dei grossi centri meccanografici, risulta piu' semplice e piu' breve (confronto COBOL/BASIC).
In particolare questa routine che proponiamo consente l'espansione in lettere di tutti i numeri compresi tra lo -zero- e -novecento novanta nove miliardi novecento novanta nove milioni novecento novanta nove mila novecento novanta nove-

(pausa per tirare il respiro).

Si e' cercato di utilizzare il minor numero di DATA, sfruttando la possibilita' di combinare opportunamente determinati elementi fondamentali contenuti nella subroutine (righe 420-550).
Il problema, infatti, e' di poter 'contare' fino a 999; il numero 999.999.999.999, ad esempio, e' composto dalla parola
-novecentonovantanove-
ripetuta quattro volte, con la semplice aggiunta delle parole -miliardi-, -milioni- e -mila-.

```
10 REM ****************************
15 REM *** ESPANSIONE  NUMERI ***
20 REM *** ROBERTO    SOZZANI ***
30 REM *** POCKET GROUP  1981 ***
40 REM ****************************
50 PRINT "□"
60 DIM UU$(19)
70 FOR I=1 TO 19: READ UU$(I): NEXT
90 FOR I=1 TO 9: READ DD$(I): NEXT
100 FOR I=1 TO 9: READ CC$(I): NEXT
120 U=0: D=0: C=0
130 INPUT "▒▒▒NUMERO■ "; X$
135 IF X$="0" THEN PRINT"▒ZERO": GOTO 130
140 LX=LEN(X$): IF LX=12 THEN XX$=X$: GOTO 180
150 IF LX>12 THEN 130
160 Q$="": FOR J=1 TO 12-LX: Q$=Q$+" ": NEXT
170 XX$=Q$+X$
180 X0$=MID$(XX$,10,3)
190 X1$=MID$(XX$,7,3)
200 X2$=MID$(XX$,4,3)
210 X3$=MID$(XX$,1,3)
220 A=VAL(X3$): GOSUB420: MD$=F$
240 A=VAL(X2$): GOSUB420: ML$=F$
260 A=VAL(X1$): GOSUB420: MG$=F$
280 A=VAL(X0$): GOSUB420: MC$=F$
300 IF MD$="▮UNO" THEN MD$="UNMILIARDO": GOTO340
310 IF MD$="▮OTTO" THEN MD$=" "+MD$
320 IF VAL(X3$)>0 THEN MD$=MD$+"MILIARDI"
330 IF ML$="▮OTTO" THEN ML$=" "+ML$
340 IF ML$="▮UNO" THEN ML$="UNMILIONE":GOTO 360
350 IF VAL(X2$)>0 THEN ML$=ML$+"MILIONI"
360 IF MG$="▮UNO" THEN MG$="MILLE":GOTO 390
370 IF MG$="▮OTTO" THEN MG$=" "+MG$
380 IF VAL(X1$)>0 THEN MG$=MG$+"MILA"
390 IF MC$="▮UNO" THEN MC$=" "+MC$
400 IF MC$="▮OTTO" THEN MC$=" "+MC$
410 PRINT: PRINT MD$+ML$+MG$+MC$: GOTO 120
420 U=0: D=0: C=0: F$=""
425 IF A=0 THEN RETURN
430 IF A<20 THEN NU$=UU$(A): GOTO 500
440 LA=LEN(STR$(A))
450 U$=RIGHT$(STR$(A),2): U=VAL(U$)
455 IF U<20 THEN 480
460 U$=RIGHT$(STR$(A),1): U=VAL(U$)
470 D$=MID$(STR$(A),LA-1,1): D=VAL(D$)
480 C$=MID$(STR$(A),LA-2,1): C=VAL(C$)
490 NU$=UU$(U)
500 ND$=DD$(D)+NU$
510 IF D=0 AND U=1 THEN 540
520 IF D=0 AND U=8 THEN 540
530 NC$=CC$(C)+ND$: GOTO550
540 NC$=CC$(C)+" "+NU$
550 F$=NC$: RETURN
560 DATA"▮UNO",DUE,TRE,QUATTRO,CINQUE,SEI,SETTE
565 DATA"▮OTTO",NOVE,DIECI,UNDICI,DODICI,TREDICI
570 DATAQUATTORDICI,QUINDICI,SEDICI,DICIASSETTE
580 DATADICIOTTO,DICIANNOVE
590 DATADIECI,VENTI,TRENTA,QUARANTA,CINQUANTA
600 DATASESSANTA,SETTANTA,OTTANTA,NOVANTA
610 DATACENTO,DUECENTO,TRECENTO
615 DATAQUATTROCENTO,CINQUECENTO,SEICENTO
620 DATASETTECENTO,OTTOCENTO,NOVECENTO
```

Il programma puo' essere scomposto in cinque parti principali:

righe	commento
60-110	Lettura ed inserimento in tabelle dei DATA, per un totale di trentasette elementi.
130-210	Input della variabile X$ e trasformazione della stessa in una variabile di dodici elementi con successiva scomposizione in quattro sottovariabili (X0$, X1$, X2$, X3$), di tre cifre ciascuna, che verranno analizzate dalla subroutine.
300-410	Trasformazione della variabile in OUTPUT, con l'eventuale aggiunta delle parole -miliardi, milioni- ecc. rispettando l'esatta sintassi (in particolare i numeri uno e otto creano qualche problema), e stampa finale.
420-550	Subroutine in grado di 'leggere' fino al numero 999, e cuore del programma stesso. Ogni gruppo di tre cifre viene qui analizzato per subire una prima espansione.
560-620	DATA

Queste cinque parti potranno essere inserite in un programma di stampa che necessiti in un momento dell'esecuzione l'espansione di un numero.

Non pensiamo di dover spiegare riga per riga questa routine; riteniamo che chi abbia una sia pur minima conoscenza di programmazione in BASIC possa capirne il funzionamento dopo una breve lettura delle varie istruzioni che la compongono.

---**H**--- ---**H**--- ---**H**---

Inversione schermo

di Roberto Sozzani

```
1 REM ********************************
2 REM ** ROUTINE INVERSIONE SCHERMO **
3 REM ********************************
4 FOR A=826 TO 858
5 READ B
6 POKE A,B
7 NEXT
8 DATA 169,128,141,74,3,141,77,3,160
9 DATA 4,162,0,169,128,93,0,128,157
10 DATA 0,128,232,208,245,238,74,3,238
11 DATA 77,3,136,208,236,96
12 A=0:B=0
```

Questa breve routine si rivela utile in tutti quei casi in cui risulti necessario "reversare" lo schermo.
Che cosa si intende con il termine "reversare" ? E' presto detto:
Sappaimo oramai tutti a cosa serve il tasto di RVS e OFF-RVS, il tasto cioe' che ci permette di porre una stringa alfanumerica su video in negativo o in positivo.
Ma se volessimo porre tutto lo schermo video in negativo, tutto ed in un colpo solo, in un batter di ciglia, la faccenda diventa impossibile se si volesse utilizzare il linguaggio BASIC.
Entra quindi in gioco il linguaggio macchina che ha proprio la prerogativa della velocita e la sintesi di istruzioni.
Non voglio in queste righe intrattenervi con complicate spiegazioni delle istruzioni del linguaggio macchina, ma accettiamo cosi' come e' la routine ed utiliziamola nel migliore dei modi.
Una volta caricata, il vero e proprio programma in LM risiede in una zona di memoria che normalmente non viene toccato da altri programmi ó seguenti caricamenti.
Un comando SYS(826) e' sufficente per ottenere l'inversione di tutto cio' che e' presente in quel momento sullo schermo.
Allora con semplici istruzioni e' possibile far lampeggiare lo schermo quante volte vogliamo o necessitiamo al fine di ottenere efficaci effetti visivi.
Penso, d'altro canto che un "PETtarolo" degno di questo nome non abbia bisogno di suggerimenti su come poter utilizzare questa routine.
Le righe proposte possono essere naturalmente inserite in un qualsiasi programma. Per questo scopo si carica come primo passo la routine e quindi si potra' proseguire con lo scrivere il proprio programma cominciando, come e' conseitudine, dalla riga 100.

Vediamo riga per riga cosa succede:

Riga 1,2 e 3 : hanno il solo scopo estetico di presentazione.

Riga 4 : inizia il ciclo di FOR...NEXT che parte proprio con un numero corrispondente alla locazione di memoria 826. Come e' stato gia' detto la routine risiedera' proprio in una locazione di memoria che normalmente non viene utilizzata, quella relativa al buffer della seconda unita' a cassetta, che appunto inizia dalla 826. Le celle di

memoria necessarie saranno solamente 33 e quindi il numero corrispondente alla cella finale sara' proprio 858.
Ricordo ancora una volta che una volta caricata, questa routine resta in memoria fino a quando non si spegne il PET o fino a quando non si utilizzi il secondo buffer per altre necessita'.

Riga 5 : Comando BASIC di lettura dei dati riportati nelle righe DATA.

Riga 6 : questo comando provvede a sistemare i singoli valori man mano letti nel buffer. La prima volta sara': POKE 826,196 ; poi POKE 827,128 .... fino a POKE 858,96.

Riga 7 : chiusure del ciclo FOR...NEXT.

Righe 8,9,10 e 11 : DATA. E' l'elenco completo delle istruzioni in linguaggio macchina espresse in assoluto che ci permetteranno l'inversione dello schermo.

Riga 12 : Alla fine vengono azzarati i valori contenuti nelle variabili A e B. Questa precauzione risulta necessaria solamente se le medesime variabili siano presenti nei programmi ai quali si aggiunge la routine.

Ma vediamo come si usa in pratica.
La semplicita' d'uso e' addirittura banale: una volta eseguito il comando di RUN la routine viene memorizzata e quindi sara' sufficente inserire nel punto prestabilito del programma il comando SYS(826) per ottenere come risultato l'inversione dllo schermo.
Ad esempio:

```
100 SYS(826)
110 FOR I=1 TO 500
120 NEXT
130 SYS(826)
```

Il risultato che otterremo con queste istruzioni sara' quello di vedere lo schermo per circa mezzo secondo in "reverse".
Un altro effetto potra' essere quello di far lampeggiare il video per dieci volte. Vediamo come:

```
100 FOR K=1 TO 20
110 SYS(826)
120 FOR I=1 TO 500
130 NEXT I,K
```

Dove la variabile K conterra' un valore massimo pari al doppio del numero dei lampeggi che vorremo ottenere e dove la variabile I serve per regolare la velocita' del lampo.

Che altro dire ? Sbizzarritevi !

---**H**--- ---**H**--- ---**H**---

Il latte fa bene
ovvero
La gestione delle stalle da latte

Molti di noi si ricordano questo famoso slogan legato alla famosa Anita.
Sono andato a visitare una aziende produttrice di latte ed ero partito con l'immagine nella mente di quelle stalle che da bambino avevo visto in campagna. Casolare, aia, polli, anatre, cane e stalla piuttosto odorosa... si insomma un ambiente contadino come esisteva una volta e che oggi non e' facile trovare.
Oggi le cose sono cambiate, le mucche, daccordo sono ancora fatte di carne ed ossa, ma tutto l'ambiente ha un colore diverso, moderno ed e' proprio una parte di questa modernita' che sono andato a toccare con mano.
La mia visita era dettata dall'interesse di veder il sistema PET-CBM che funzionava con una procedura, presentata allo scorso SMAU, che provvedeva alla gestione automatica di una stalla da latte.
Gli scopi principali che la procedura si prefigge non riguardano esclusivamente la produzione del latte, ma anzi segue una situazione generale tanto da tenere sotto controllo ogni capo di bestiame con ben 56 dati di base (codice, nascita, gravidanze, produzione ecc.).
Il pacchetto di programmi e' in grado di gestire in tempo reale tutti questi dati e fornisce innumerevoli ed utili informazioni.
Il costo generale del sistema risulta, come per altre applicazioni, estremamente basso tanto che l'approccio uomo, con le sue necessita', e computer e' veramente a portata di ognuno.
Questa procedura e' stata studiata da uno dei gruppi software legati alla Harden e prevede una configurazione hardware composta dal PET-CBM 3032 con floppy disk 3040 e stampante 8024 e si adatta ad una vasta fascia di utilizzatori: dalle associazioni di agricoltori ai coldiretti e a tutti gli operatori del settore.

PEEK & POKE

Sul numero zero di Pocket PET abbiamo roportato alcuni interessanti comandi di PEEK, POKE e di SYS.
Vogliamo continuare ancora su questo filone proponendo altre soluzioni del medesimo tipo e... vediamo....

Cosa accade se :

comando	risultato
PRINT PEEK (50003).......	1 se il PET in possesso e' provvisto di nuove ROM's 0 se il PET in possesso e' provvisto di vecchie ROM's
POKE 59458,62............	velocizza l'edit di schermo, aumentando la velocita' normale di ben 3 volte. E' da tenere presente che nei nuovi modelli del PET/CBM serie 3000 questa POKE non puo' usata in quanto questa locazione di memoria non prevede questo comando. Se si volesse tentare ugualmente il risultato sarebbe quello del blocco completo del sistema.
POKE 59458,30............	ripristina la velocita di edit di schermo al valore nominale
SYS (64721)..............	System reset (questo comando simula lo spegnimento e la riaccensione del computer) (per i PET con vecchie ROM's per ottenere il medesimo risultato si deve eseguire SYS (64824))
SYS (57867)..............	genera un comando equivalente al CLEAR SCREEN
SYS (57943)..............	genera un comando equivalente al HOME CURSOR
SYS (1024)...............	abilita il sistema MONITOR (TIM) solo per PET con nuove ROM's.
SYS (4)..................	per semplicita' questo comando ottiene il medesimo risultato di quello predente. Scegliete quello che piu' vi piace.

Un carattere sullo schermo

La scrittura su schermo di un carattere qualunque del set di caratteri ASCII o simboli grafici, in una locazione specificata dall'utente, si puo' fare con una semplice formuletta che elenchiamo qui' di seguito:

```
POKE A+((Y-1)*40)+X-1,C
```

Si puo' impiegare questa formuletta anche come subroutine; per far questo si aggiungera' -:RETURN- dopo la lettera C della POKE, tenendo pero' presente che bisogna prima definire i parametri A, Y, X e C al fine di ottenere un giusto edit sullo schermo.

Parametro A = Prima locazione della memoria di schermo (32768).
Parametro Y = Riga che si vuole stampare il carattere desiderato.
Parametro X = Colonna che si vuole stampare il carattere desiderato.
Parametro C = Valore ASCII del carattere o simbolo grafico che si vuole stampare.

Si puo' anche definire il valore di C usando il seguente metodo: C=ASC("*"); in questo semplice esempio il carattere asterisco verra' stampato sul video nel punto specificato in precedenza dalle variabili Y e X.

---**H**---

Pagina Zero

Il microprocessore 6502, che viene utilizzato dal PET/CBM, utilizza il primo K di memoria per particolari funzioni richiamate dal sistema operativo.
Queste prime 1024 celle di memoria si suddividono in quattro parti, chiamate pagine. Ogni pagina quindi si divide in 256 celle di memoria.
La 6502 controlla, nel PET/CBM, 256 pagine; dalla pagina 00 alla pagina FF.
256 pagine per 256 celle di memoria uguale 65536; cifra che corrisponde esattamente ai 64K (64 x 1024) che la 6502 puo' controllare. I primi 32K costituiscono la memoria RAM (parte dedicata all'utente), mentre i seguenti 32K sono dedicati al sistema operativo Firmware (ROM).
In questo numero di Pocket PET abbiamo pensato interessante riportare per intero tutta la pagina zero di memoria e quindi sui prossimi numeri pubblicheremo quelle altre pagine che potranno essere di sicuro aiuto per chi soprattutto si vorra' addentrare nel campo sofisticato vicino al linguaggio macchina.

Conoscete la

HARDEN S.p.A.

La società italiana che importa in esclusiva per voi tutti i prodotti della

commodore

? ? ?

Certo che si!!!
Perché voi siete possessori di PET/CBM o siete in procinto di accapararvene uno.
Allora inevitabilmente dovete conoscere questo nuovo nato in casa

e... dovrete abbonarvici.

La nuova rivista:

POCKET PET

è per tutti voi! ! !

Pocket PET anno 1 numero 0 come abbonarsi?

Negli Stati Uniti e poi anche in Inghilterra ed in qualche altro paese dopo la diffusione del **PET/CBM** si è venuta a creare la necessità di redigere dei fascicoli dattiloscritti la cui diffusione avviene tramite abbonamento.
Non poteva mancare anche qui in Italia un qualche cosa di similare.
La Harden s.p.a. ha fatto propria questa necessità e senza alcun indugio ha dato il varo a questa sua nuova "produzione" che nulla a che vedere con un ciclostilato.
Nasce allora una nuova rivista. Una rivista per tutti i possessori e simpatizzanti del **PET/CBM.**
L'impegno finanziario della Harden s.p.a. non si ferma solamente alla produzione del **Pocket PET**, ma al contrario prevede ben tre formule di abbonamento alla rivista. Vediamole insieme:

a) abbonamento a prezzo intero per la durata di un anno (6 numeri).
Questa formula è stata prevista per tutti coloro che non siano possessori di PET/CBM Harden-Commodore. Quota L. 10.000

b) abbonamento per la durata di un anno (6 numeri) con sconto speciale del 50%.
A questa forma di abbonamento hanno diritto di aderire tutti coloro che siano diventati possessori di PET/CBM Harden-Commodore prima del 1-1-81.
Quota L. 5.000

c) abbonamento per la durata di un anno (6 numeri) con sconto speciale del 100% per tutti coloro che diventano possessori di PET/CBM Harden-Commodore nell'arco dell'anno in corso, il 1981.
Quota L. gratis

Nella pagina a fianco è stato predisposto uno speciale questionario/cedola di abbonamento che dovrà essere compilato a cura del futuro abbonato.

Per chi deve scegliere la forma di abbonamento **a** dovrà essere compilata la parte 1 ed eventualmentele parti **2** e **3**.

Per chi potrà usufruire della formula **b** sarà necessario compilare i riquadri **1**, **2** e **4** ed eventualmente quello contrassegnato con il numero **3**.

Per tutti i nuovi acquirenti, quelli cioè che potranno usufruire dello sconto del 100% (formula **c**), dovranno essere compilati i riquadri **1**, **2**, eventualmente il riquadro **3** ed infine, -conditio sine qua non- perché questa forma di abbonamento possa risultare valida a tutti gli effetti, dovrà essere compilata da parte del rivenditore autorizzato Harden la parte contrassegnata dal numero **5**.

Dopo aver controllato che ogni parte richiesta sia stata compilata in modo chiaro e completo sarà sufficente piegare, graffare e spedire o consegnare al proprio rivenditore Harden di fiducia, e.....

benvenuti nella grande famiglia
Harden-Commodore

1

nome ....

via ....

cap .... località ....

formula di abbonamento a ☐ b ☐ c ☐

solamente per la formula di abbonamento **a** o **b**

☐ ho effettuato il versamento di L. .... tramite bollettino di conto corrente postale numero 17/9993 intestato alla Harden spa Sospiro (Cremona).

☐ includo assegno bancario relativo all'importo di L. .... intestato alla Harden spa Sospiro (Cremona)

Possiedi un PET/CBM?

2

			acquisterò	
mod.	K RAM	serie n.	si	no

---

Di quali periferiche disponi?

3

unità cassetta C2N		serie n.	si	no
seconda unità a cassetta C2N		serie n.	si	no
stampante	mod.	serie n.	si	no
unità floppy disk	mod.	serie n.	si	no
plotter	mod.	serie n.	si	no
modem acustico	mod.	serie n.	si	no
tavoletta grafica	mod.	serie n.	si	no

---

4

data approsimativa di acquisto: mese .... anno ....

presso il mio rivenditore di fiducia:

riv. ....

via ....

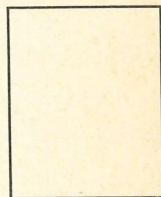
cap. .... località ....

5

parte riservata al rivenditore

data di acquisto

timbro e firma

HARDEN S.p.A.
26048 SOSPIRO (CREMONA)
ITALIA

RAM Soubroutines

ROMs nuove	ROMs vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
112 135	194 199	230	230	routin riporto nuovo carattere BASIC
118	200	173	173	riferimento dell'attuale carattere
119 120	201 202	1904	1929	puntatore di testo sorgente
136 140	203 223	128	201	area di work per numerazione random

Operation System Zero page

ROMs nuove	ROMs vecchie	semplice valore nuove	semplice valore vecchie	Descrizione
141 143	512 514	398710	3831352	24 ore clock 1/60second
144 145	537 538	58926		test vet. HW (test key STOP (228.225)
146 147	539 540	64791	0	6502 BRK instruction interrupt vector
148 149		50057		NMI interrup vector
150	524	0	0	byte di stato di I/O
151	515	255	255	immagine ultimo tasto premuto
152	516	0	0	immagine tasto SHIFT (1=premuto)
153 154	517 518	65282	37916	correzione del fattore di clock
155	521	255	255	copia del tasto ultima riga
156		0		buffer costante di tempo
157	523	0	0	flag I/O (0=LOAD)(1=VERIFY)
158	525	0	0	numero di char. nel buffer tastiera
159	526	0	0	flag di reverse video
160		0		IEEE 488 flag di output
161	542	13	13	carattere nella riga dello schermo
162		0		utility
163 164	544 545	11-13	13	routine di input del cursore
165		63		IEEE 488 buffer char in output
166	547	255	255	codice input della tastiera
167	548	1	1	flag di abilitazione cursore
168	549	17	11	flag di temporizzazione del cursore
169	550	32	32	valore del char. di input sul video
170	551	0	0	cursore ON/OFF
171	552	0	0	EOT bit ricevitore, tape write
172	608	0	0	flag di input (0=tastiera 1=video)
173		0		utility
174	610	1	1	numero dei file aperti
175	611	0	0	periferica di INPUT (0=tastiera)
176	612	3	3	periferica di OUTPUT (3=video)
177	613	0	0	byte di parita per tc
178		0		flag per byte ricevente
179		0		utility
180		0		buffer dei char. tc
181	616	0	0	puntatore per trasferimento filename
182				I/O utility
183	620	0	8	contatore seriale
184	622	0	0	utility per tc
185	623	0	0	contatore ciclico per tc
186	624	0	0	contatore sincronizzato per tc/write
187	625	0	0	ind.prossimo char. I/O nel buffer tc1
188	626	0	0	ind.próssimo char. I/O nel buffer tc2
189	627	0	0	contatore sincronizzato per tc/read
190	628	0	0	flag per indicare errore tc bit/byte
191	629	0	0	flag di errore di tc
192	630	0	0	flag di errore di tc
193	631	0	0	contatore per tape cassette
194	632	0	128	flag di funzione dell'attuale tc/read
195	633	0	9	utility di parita'
196 197	224 225	33728	0	posizione della linea sullo schermo
198	226	0	0	posizione del cursore sulla linea
199 200	227 228	33792	0	indirizzo di partenza
201 202	229 230	0	0	ind. fine prog.in caricamento
203 204	231 232	0	0	costante di tempo tape tassette (tc)
205	234	0	0	flag di aperte virgolette (")
206	235	0	0	timer 1 di status di interrupt
207	236	0	0	ricevitore di char. di EOT
208	237	0	0	ricevitore di carattere di errore
209	238	0	0	lunghezza dell'attuale file-name
210	239	4	5	numero dell'attuale file logico
211	240	255	255	attuale indirizzo secondario
212	241	4	63	numero dell'attuale device
213	242	39	39	lunghezza massima della riga video
214 215	243 244	0	634	puntatore dello start del buffer tc
216	245	24	23	num.riga dove il cursore si trova
217	246	10	10	puntatore di I/O buffer
218 219	247 248	0	1024	puntatore loc. inizio per VERIFY-LOAD
220	251	0	0	num. del tasto inserito
221	252	0	9	serial bit shift word
222	253	0	0	num. di blocchi che rimangono (R/W)
223	254	0	0	puntatore del buffer della parola
224 248	553 577			tab. di LSB di start linee del video
249	519	0	0	interrupt drive flag tape cassette 1
250	520	0	0	interrupt drive flag tape cassette 2
251 252		54144		indirizzo di start cassette
253 255		243		utility

Locazione di funzione di USR

ROMs nuove	vecchie	semplice valore nuove	vecchie	Descrizione
0	0	76	76	costante per 6502 per istruzione JMP
1 2	1 2	826	826	indirizzo del vettore di salto

Evoluzione delle variabili e Terminal I/O Maintenance

ROMs nuove	vecchie	semplice valore nuove	vecchie	Descrizione
3		0		ricerca carattere
4		0		delimitatore di modo apici (")
5	92	255	255	puntatore buffer input, count.gen.
6	93	0	0	flag di dimensionamento variabi
7	94	0	0	flag di tipo di variabile (0=n FF=a)
8	95	0	0	flag di var.num (0=floating FF=int)
9		0		flag di scans. DATA. LIST. Memoria
10	97	0	0	flag di variabile. Flag di FNx
11	98	0	0	tipo input (0=INPUT 64=GET 152=READ)
12		0		flag per segno ATN
13	100	0	0	flag per output suppress (+=n -=s)
14		0		attuale canale I/O
15		40		inutilizzato
16		30		limite di scansione (inutilizzato)
17 18		828		indirizzo BASIC per SYS o GOTO
19	101	22	104	indirizzo della prossima variabile
20 21	102 103	19	101	puntatore dell'ultima tringa tempor.
22 29	104 111	2	2	tabella di puntamento variabile
30 31	112 113	16451	14525	indice indiretto #1
32 33	114 115	26119	62983	indice indiretto #2
34	116	1	1	pseudo registro di operatore di funz.
35	117	140	234	
36	118	0	0	
37	119	0	0	
38	120	0	0	
39	121	0	0	

Data Storage Maintenance

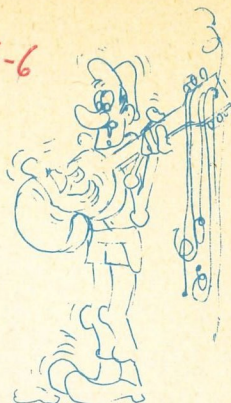
ROMs nuove	vecchie	semplice valore nuove	vecchie	Descrizione
40 41	122 123	1025	1025	puntatore di partenza BASIC
42 43	124 125	1920	1946	puntatore di partenza variabili
44 45	126 127	2032	2072	puntatore di partenza tabelle
46 47	128 129	2191	2231	puntatore di inizio zona libera
48 49	130 131	8192	8192	puntatore di partenza della stringa
50 51	132 133	8191	8191	puntatore di fine della stringa
52 53	134 135	8192	8192	indirizzo di fine RAM
54 55	136 137	2000	2000	numero della riga BASIC attuale
56 57	138 139	110	110	n. riga precedente per eventale CONT
58 59	140 141	1897	1922	puntatore della prossima riga
60 61	142 143	200	1150	puntatore dell'attuale linea di DATA
62 63	144 145	1855	1879	puntatore dell'attuale DATA

Evoluzione di espressioni

ROMs nuove	vecchie	semplice valore nuove	vecchie	Descrizione
64 65	146 147	514	13	vettore dell'INPUT
66 67	148 149	89	89	nome della variabile attuale
68 69	150 151	2006	2032	puntatore di partenza della variabile
70 71	152 153	2006	2032	puntatore di riferimento per FOR.NEXT
72 73	154 155	1279	31999	puntatore della tabella
74	156	0	0	maschera dell'operatore attuale
75 76	157 158	62268	898	puntatore di definizione di funzione
77 78	159 160	26531	104	puntatore di descrizione di stringa
79	161	243	221	lunghezza della stringa
80	162	3	3	costante di collegamento
81	163	76	76	costante per 6502 per istruzione JMP
82 83	164 165	0	0	vettore di funzione
84 89	166 171	211	129	floating point accumulator #3
90 91	172 173	0	0	pointer del blocco da trasferire #1
92 93	174 175	0	0	pointer del blocco da trasferire #2
94 99	176 181	0	0	floating point accumulator #1
100	182	0	0	copia del segno di frazione di FAC #1
101	183	0	0	contatore per FAC #1
102 107	184 189	0	0	floating point accumulator #2
108	190	0	0	owerflow per floating
109	191	0	0	copia del segno di frazione di FAC #2
110 111	192 193	258	258	pointer di conversione

Effetti sonori

PET & CBM 2001/3032

di Massimo Rossi

Sul numero zero Pocket PET, abbiamo accennato come poter mettere il vostro computer in grado di produrre suoni.
Se la cosa vi ha interessato, avrete, senza dubbio, provato a collegare il vostro PET/CBM ad un amplificatore, tramite i piedini **M** e **N** della USER-PORT.

Se tutto e' andato bene, dall'altoparlante del vostro amplificatore, sara' uscito un suono: la voce del vostro computer.
Se al contrario avete avuto qualche difficolta', allora, di certo, la Harden, presso tutti i suoi rivenditori autorizzati, ha quello che fa per voi: un amplificatore studiato appositamente per il PET/CBM; dotato di interruttore, di spina di alimentazione, di connettore speciale per la USER-PORT, di spia luminosa di ON-OFF e di regolatore di volume (non piu' bambini svegli di notte, mogli o madri iraconde.. !). Il tutto e' racchiuso in un elegante contenitore di colore nero, denominato **MUSIC BOX**.

Per i PETaroli che si interessano di musica, e che, comunque, vorrebbero dotare i loro programmi di effetti sonori o musicali, abbiamo creato questo semplice programmino, che li mettera' in grado di creare l'effetto da loro desiderato, e fornira' loro, l'esatta collocazione dei POKE interessati.
Coloro che hanno gia' seguito sul numero zero di Pocket PET, la rubrica PEEK & POKE, ci vorranno scusare, se ripetiamo, a brevi linee, come si ottiene la produzione di suoni dal PET/CBM.
Alcuni piedini della USER-PORT, sono in grado di fornire una tensione di +5 Volt, su richiesta del programma in BASIC.

Un piedino in particolare fornisce un segnale ad onda quadra, modulabile in frequenza.
Quali sono, allora, i comandi che abilitano questa uscita, e quali ne regolano la frequenza di emissione?
E' presto detto:

```
POKE 59467,16  abilita il modo musica.
POKE 59466,Y   decide il range di
               frequenza.
               (Y, da 1 a 255)
POKE 59464,X   decide la frequenza
               (X, da 1 a 255)
```

E' molto importante ricordare che, se non si chiude il programma con una disabilitazione della USER-PORT, il computer non potra' registrare il programma, e voi rischierete di doverlo digitare di nuovo.

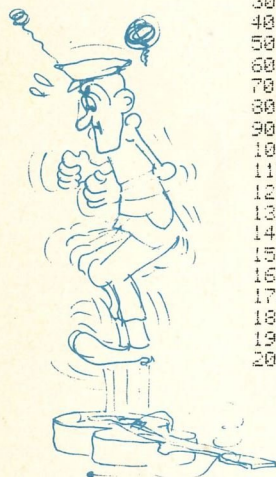
```
Quindi per aprire:
POKE 59467,16
POKE 59466,Y
POKE 59464,X
Per chiudere:
POKE 59467,0
POKE 59466,0
POKE 59464,0
```

Il programma che vi proponiamo, vi domanda in quale range di frequenza (MODE) volete lavorare. Una volta fornito questo dato (un numero compreso fra 1 e 255), apparira' sullo schermo la sequenza dei valori dei POKE, che vi fornisce la nota che udirete uscire dall'altoparlante.

Se desidarate cambiarla, tenete premuto il:
tasto + per aumentare la frequenza
tasto — per diminuire la frequenza.

Il valore del POKE digitato sullo schermo, viene aggiornato simultaneamente. Rilasciando il tasto premuto, la sequenza delle note, si ferma e si stabilizza sull'ultima nota prescelta.
Se non volete terminare l'ascolto, premete il tasto RETURN, tutto si spegnera' automaticamente, ma potrete riprendere l'ascolto, in aumento o in diminuzione, a piacere, premere di nuovo il tasto + o il tasto —.
Ricordiamo che se volete interrompere il programma, dovrete, necessariamente, interrompere la nota con il tasto RETURN poi bloccare con il tasto STOP.
Questo programma potra' esservi utile non solo per trovare gli indirizzi di POKE corrispondenti alle note che desiderate, (basta confrontare l'uscita sonora con qualsiasi strumento musicale), ma anche per sperimentare gli effetti che si possono ottenere cambiando il range di frequenza.
Sta a voi ora scoprire nuovi effetti.

REMarks.

- 10 azzeramento delle variabili
- 20 richiesta del MODE
- 30 richiesta di premere un tasto
- 40 apertura USER-PORT
- 50-70 test di tasto premuto: viene stabilito quale tasto e' stato premuto e viene incrementata o decrementata la serie agendo sul valore di N
- 80-90 condizione di inizio e fine della serie: viene mantenuta la progressione all'interno dei valori compresi fra 1 e 255
- 100 produzione della nota
- 110-140 edit del valore di POKE
- 150 se nessun tasto e' premuto, ripete la nota precedente, e riedita la pagina precedente
- 170-190 se il tasto RETURN (CHR$(13)) e' premuto, chiude la USER-PORT.

```
10 PRINT"[CLR]":N=0
20 INPUT"[DOWN]QUALE MODE VUOI";P
30 GETA$:IFA$=""THEN40
40 POKE59467,16:POKE59466,P
50 IFPEEK(151)=17THENN=N+1
60 IFPEEK(151)=9THENN=N-1
70 IFPEEK(151)=27THEN170
80 IFN<1THENN=1
90 IFN>255THENN=255
100 POKE59464,(256-N)
110 PRINT"[HOME][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN]POKE 59467,16"
120 PRINT"[DOWN]POKE 59466,";P
130 PRINT"[DOWN]                          "
140 PRINT"[UP]POKE 59464,";256-N
150 IFPEEK(151)=255THEN100
160 GOTO50
170 POKE59467,0:POKE59466,0:POKE59464,0
180 GETA$:IFA$=""THEN180
190 GOTO40
200 END
```

Per concludere questo breve articolo "rumoroso" proponiamo alcune routines che potranno essere inserite, previa rinnumerazione, in un qualsiasi programma onde abbellirne di effetti sonori adeguati alle svariate esigenze di gioco.

```
100 REM*****************************
110 REM***   'MAZZA LA NONNA...    ***
120 REM*****************************
130 P=59464
140 POKE59467,16
150 POKE59466,15:T=3
160 POKEP,188:FORX=1TO200:NEXT
170 POKEP,251:FORX=1TO100:NEXT:GOSUB250
180 POKEP,251:FORX=1TO100:NEXT
190 POKEP,225:FORX=1TO200:NEXT
200 POKEP,251:FORX=1TO300:NEXT
210 POKEP,0:FORX=1TO150:NEXT
220 POKEP,199:FORX=1TO200:NEXT:T=50:GOSUB250
230 POKEP,188:FORX=1TO150:NEXT
240 GOTO270
250 POKE59464,0:FORX=1TOT:NEXT:RETURN
260 POKE59467,0:POKE59466,0:POKE59464,0
270 POKE59467,0:POKE59466,0:POKE59464,0
280 RETURN
```

```
100 REM*****************
110 REM***   DOOINGG   ***
120 REM*****************
130 POKE59467,16
140 POKE59466,30
150 FORWZ=1TO20
160 POKE59464,WZ
170 NEXT
180 POKE59467,0
190 RETURN
```

```
100 REM*********************
110 REM***    GOCCIOLINA   ***
120 REM*********************
130 POKE59467,16
140 POKE59466,45
150 FORWX=1TO255STEP2
160 POKE59464,WX
170 NEXT
180 POKE59467,0
190 RETURN
```

```
100 REM******************************
110 REM***   SCALA DI PIANOFORTE   ***
120 REM******************************
130 POKE59467,16
140 POKE59466,15
150 FORR=0TO100STEP5
160 POKE59464,R:X=TAN(R):NEXT
170 FORR=99TO0STEP-5
180 POKE59464,R:X=TAN(R):NEXT
190 POKE59467,0:POKE59466,0:POKE59464,0
200 RETURN
```

```
100 REM*******************
110 REM***   CAMPANELLO   ***
120 REM*******************
130 P=59464
140 POKE59467,16
150 POKE59466,9
160 FORL=1TO50:POKEP,238:POKEP,251:NEXT
170 POKE59467,0:POKE59466,0:POKE59464,0
180 RETURN
```

```
100 REM********************
110 REM***   PING PONG    ***
120 REM********************
130 P=59464
140 POKE59467,16
150 POKE59466,15
160 FORJ=1TO5
170 POKEP,255:FORL=1TO60:NEXT
180 POKEP,0:FORL=1TO100↑RND(1)*20:NEXT
190 POKEP,128:FORL=1TO60:NEXT
200 POKEP,0:FORL=1TO100↑RND(1)*20:NEXT
210 FORX=1TO100:NEXT
220 NEXT
230 POKE59467,16
240 POKE59466,63
250 POKE P,255:FOR L=1 TO 500:NEXT L
260 POKE59467,0:POKE59466,0:POKE59464,0
270 RETURN
```

```
100 REM*****************************
110 REM***   DESTRA E SINISTRA   ***
120 REM*****************************
130 INPUT"TEMPO";D
140 IFD>5THEN130
150 POKE59467,16
160 POKE59466,91
170 NN=50*D
180 FORWZ=1TO20
190 POKE59464,NN
200 POKE59464,WZ
210 FORI=1TOD*100:NEXT
220 POKE59464,WZ
230 NEXT
240 POKE59467,0
250 RETURN
```

```
100 REM********************
110 REM***   FISCHIACCIO   ***
120 REM********************
130 POKE59467,16
140 POKE59466,13
150 FORR=185TO80STEP-3:POKE59464,R:NEXT
160 POKE59464,0:FORL=1TO200:NEXT
170 FORR=205TO105STEP-3:POKE59464,R:NEXT
180 FORR=105TO255STEP3:POKE59464,R:NEXT
190 POKE59467,0:POKE59466,0:POKE59464,0
200 RETURN
```

```
100 REM***********************
110 REM***   BOIIINNNGG   ***
120 REM***********************
130 POKE59467,16
140 POKE59466,146
150 FORI=1TO10
160 FORWY=100TO140STEP5
170 POKE59464,WY
180 NEXTWY
190 FORWY=140TO100STEP-5
200 POKE59464,WY
210 NEXTWY
220 POKE59464,0
230 FORX=1TO100:NEXT
240 NEXT
250 RETURN
```

```
100 REM***********************
110 REM***   CARICAA!!!   ***
120 REM***********************
130 POKE59467,16
140 POKE59466,15:T=3
150 POKEP,255:FORX=1TO100:NEXT:GOSUB230
160 POKEP,191:FORX=1TO100:NEXT:GOSUB230
170 POKEP,152:FORX=1TO100:NEXT:GOSUB230
180 POKEP,128:FORX=1TO200:NEXT:GOSUB230
190 POKEP,152:FORX=1TO100:NEXT:T=0:GOSUB230
200 POKEP,128:FORX=1TO400:NEXT
210 POKE59467,0:POKE59466,0:POKE59464,0
220 RETURN
230 POKEP,0:FORX=1TOT:NEXT:RETURN
```

```
100 REM***********************
110 REM***   GOCCIOLONA   ***
120 REM***********************
130 POKE59467,16
140 POKE59466,51
150 FORWX=1TO15
160 FORWY=180TO160STEP-2
170 POKE59464,WY
180 NEXTWY,WX
190 POKE59467,0
200 RETURN
```

```
100 REM***********************
110 REM***   IMMERSIONE!   ***
120 REM***********************
130 POKE59467,16
140 POKE59466,9
150 FORL=1TO10
160 FORR=250TO180STEP-1
170 POKE59464,R
180 NEXT:NEXT
190 POKE59467,0:POKE59466,0:POKE59464,0
200 RETURN
```

```
100 REM*******************
110 REM***   POLIZIA   ***
120 REM*******************
130 POKE59467,16
140 POKE59466,2
150 FORL=1TO4
160 FORR=100TO255STEP40
170 POKE59464,R:NEXT:FORX=1TO500:NEXT
180 FORR=255TO100STEP-40
190 POKE59464,R:NEXT:FORX=1TO500:NEXT
200 NEXT
210 RETURN
```

```
100 REM*************************
110 REM***   BOMBARDIMENTO   ***
120 REM*************************
130 POKE59467,16
140 POKE59466,85
150 FORR=50TO150:POKE59464,R:FORX=1TO30:NEXT:NEXT
160 PRINT"]           ▣ B O O M ■"
170 POKE59466,1:POKE59464,255:FORX=1TO400:NEXT
180 POKE59467,0:POKE59466,0:POKE59464,0
190 RETURN
```

```
100 REM***********************
110 REM***   HAI PERSO!   ***
120 REM***********************
130 P=59464
140 POKE59467,16
150 POKE59466,15
160 T=2
170 POKEP,237:FORX=1TO300*T:NEXT:GOSUB280
180 POKEP,237:FORX=1TO200*T:NEXT:GOSUB280
190 POKEP,237:FORX=1TO100*T:NEXT:GOSUB280
200 POKEP,237:FORX=1TO300*T:NEXT:GOSUB280
210 POKEP,199:FORX=1TO300*T:NEXT:GOSUB280
220 POKEP,211:FORX=1TO200*T:NEXT:GOSUB280
230 POKEP,237:FORX=1TO100*T:NEXT:GOSUB280
240 POKEP,237:FORX=1TO200*T:NEXT:GOSUB280
250 POKEP,251:FORX=1TO100*T:NEXT:GOSUB280
260 POKEP,237:FORX=1TO300*T:NEXT
270 POKEP,0
280 POKEP,0:FORX=1TO5:NEXT:RETURN
290 RETURN
```

```
100 REM*****************
110 REM***   LASER   ***
120 REM*****************
130 POKE59467,16
140 POKE59466,15
150 FORL=1TO5
160 FORR=0TO100STEP5
170 POKE59464,R
180 FORX=1TO2
190 NEXT:NEXT:NEXT
200 POKE59467,0
210 POKE59466,0
220 POKE59464,0
230 RETURN
```

```
100 REM********************************
110 REM***   RAGGIO DELLA MORTE    ***
120 REM********************************
130 P=59464
140 POKE59467,16
150 POKE59466,15
160 FORL=1TO200
170 POKEP,150:POKEP,200:POKEP,255
180 NEXT
190 POKEP,0:POKE59466,0:POKE59467,0
200 RETURN
```

```
100 REM***********************
110 REM***   SU´ & GIU´   ***
120 REM***********************
130 P=59464
140 POKE59467,16
150 POKE59466,85
160 FORL=1TO20
170 FORL1=1TO14:POKE59464,L1*16:NEXT
180 NEXT
190 POKE59467,0:POKE59466,0:POKE59464,0
200 RETURN
```

```
100 REM***************************
110 REM***   EFFETTO SPAZIO   ***
120 REM***************************
130 POKE59467,16
140 POKE59466,15
150 FORL=1TO200:POKE59464,10+100*RND(1)
160 FORI=1TO6:NEXT:NEXT
170 POKE59467,0:POKE59466,0:POKE59464,
180 RETURN
```

```
100 REM*********************
110 REM***   UCCELLINO   ***
120 REM*********************
130 P=59464
140 POKE59467,16
150 POKE59466,85
160 FORL=1TO10
170 POKEP,0:FORI=1TO50:NEXT
180 FORL1=152TO56STEP-8
190 POKE59464,L1:NEXT:NEXT
200 POKE59467,0:POKE59466,0:POKE59464,0
210 RETURN
```

```
100 REM************************
110 REM***   DISCO VOLANTE   ***
120 REM************************
130 P=59464
140 POKE59467,16
150 POKE59466,29
160 FORL=160TO0STEP-.3
170 POKEP,L:POKEP,L+5:POKEP,L+15
180 FORZ=1TO0:NEXT:NEXT
190 POKE59467,0:POKE59466,0:POKE59464,0
200 RETURN
```

```
100 REM************************
110 REM***   MOTOCICLETTA   ***
120 REM************************
130 P=59464
140 POKE59467,16
150 POKE59466,3
160 FORL=200TO235STEP.7:POKEP,L:POKEP,L+5:POKEP,L+20:FORZ=1TO20:NEXT:NEXT
170 FORL=1TO100:POKE59467,0:POKE59466,16:POKE59467,16:POKE59466,3:NEXT
180 FORL=235TO170STEP-.6:POKEP,L:POKEP,L+5:POKEP,L+15:FORZ=1TO0:NEXT:NEXT
190 FORL=170TO220STEP3:POKEP,L:NEXT
200 FORL=220TO129STEP-.5:POKEP,L:POKEP,L+5:POKEP,L+15:FORZ=1TO0:NEXT:NEXT
210 FORL=129TO220STEP3:POKEP,L:NEXT
220 FORL=200TO120STEP-.4:POKEP,L:POKEP,L+5:POKEP,L+15:FORZ=1TO0:NEXT:NEXT
230 FORL=120TO180STEP3:POKEP,L:NEXT
240 FORL=180TO140STEP-.2:POKEP,L:POKEP,L+5:POKEP,L+15:FORZ=1TO0:NEXT:NEXT
250 FORL=140TO200STEP2:POKEP,L:POKEP,L+5:POKEP,L+15:FORZ=1TO0:NEXT:NEXT
260 P=59464:L=200:POKE59467,16:POKE59466,51
270 FORX=1TO50:POKEP,L:POKEP,L+7:POKEP,L+14:NEXT
280 POKE59464,0
290 POKE59464,0:FORX=1TOT:NEXT
300 POKE59467,0:POKE59466,0:POKE59464,0
310 RETURN
```

---**H***---   ---**H***---   ---**H***---

PET-CBM e lo sport

di Gloriano Rossi (i2KH).

No, non e' il solito gioco di simulazione di una gara sportiva con N difficolta' od altro.
Fino alla meta dello scorso anno, le gare sportive di motociclismo e di automobilismo con prove di regolarita', sfruttavano l'oneroso utilizzo di centri meccanografici di alcuni Services al fine di avere un unico fatto positivo che consisteva nella consegna per il giorno seguente, quando andava bene, di un tabulato che riportava la graduatoria e la statistica delle prove.

A settembre del 1980 entra in campo, due giorni prima della gara, la EASY System di Cremona.

Preventivo? Addirittura redicolo al confronto delle precedenti fatture, e quindi accettato senza riserve.
Allora subito al lavoro: analisi con la consulenza dei giudici di gara e via partenza con la programmazione.
Dopo quarantotto ore insonni:

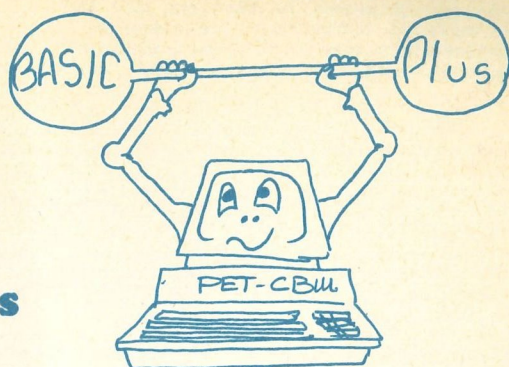
Configurazione Commodore:
CBM 3032
3040
stampante
Programmi (piuttosto grossini):
1- Inserimento dati generali e dati degli equipaggi.
2- Inserimento e variazioni dei risultati e visualizzazione real-times. Creazione di spooling.
3- Esecuzione delle stampe delle classifiche, delle statistiche secondo le necessita' della gara.

Bene, tutto funziona; si porta il sistema vicino alla giuria, (ve lo figurate un 370 della IBM o un L62 della Honeywell trasposrtato in loco?) e si incomincia a far partire il primo programma, poi al comando di START via con il secondo.
Alla fine della gara viene fatto "girare" l'ultimo programma, quello che stampa, ed ecco, come d'incanto la stampante macinare carta su carta, dati su dati, classifica generale, classifica per classe, classifica globale, classifica ....., classifica ... .

"Come gia' pronto !?!?"
"Si. Certo", e via con orgoglio al meritato riposo.

La EASY System, dopo questa positiva esperienza, ha effettuato altre assistenze quali relly o di motocross.
I risultati? Sempre positivi sotto ogni aspetto: qualita', velocita e ... prezzo.
A prova di tutto cio' sono arrivati in redazione due tabulati come esempio. Il primo relativo a quella fatidica gara, che era organizzata dall'Automobile Club di Cremona, il I trofeo CARIPLO relativo al V Rally nazionale "L.Feraboli" ed il secondo relativo all'ultima assistenza, cronologicamente cronologicamente parlando, relativa al I trofeo "A.Dossena" organizzato dal Moto Club di Crema.

Si puo' concludere questo breve redazionale con le stesse parole con cui termina un articolo, relativo ad una delle gare assistite, apparso su due colonne di un quotidiano:
"... grazie quindi alla EASY System di Cremona che ha curato l'elaborazione dei dati e delle classifiche con l'elaboratore CBM 3032 e che solamente mezzora dopo la conclusione delle varie prove aveva gia fornito le classifiche. Un vero e proprio record di efficenza che e' stato molto apprezzato da tutti".

Qualche cosa di piu' al
BASIC standard del PET/CBM

BASIC Plus

Quando, terminato il numero zero di Pocket PET, si e' iniziato ad eseguire una cernita di argomenti adatti ad essere pubblicati su questo numero, abbiamo optato, fra l'altro e senza alcun dubbio, sul BASIC Plus.

Che cosa e' in sintesi il BASIC Plus?

E' presto detto.
Il BASIC Plus e' un insieme di routinette scritte in linguaggio macchina che opportunamente richiamate svolgono delle particolari funzioni di estrema utilita', soprrattutto per chi normalmente o sporadicamente fa programmi.

Il sistema di richiamo delle varie funzioni speciali del BASIC Plus e' quello classico delle normali funzioni del BASIC standard. In ogni momento e in qualsiasi posizione dello schermo, nonche' nel sistema abbreviato, e' possibile richiamare la singola routine del BASIC Plus.
Il prodotto BASIC Plus viene venduto essenzialmente in tre versioni base, che sono:

-BASIC Plus per serie 3000 con 32KRAM da caricare ed attivare con il comando SYS(30000)

-DOS Plus per serie 3000 con 32KRAM che comprende il BASIC Plus e il DOS da caricare ed attivare con il comando SYS(30000).

-BASIC Plus per serie 3000 sia con 8 che 32 KRAM in versione circuito integrato (vedi nota a fine articolo) da inserire direttamente sulla piastra dei componenti. La sua attivazione si ottiene con il comando SYS(47100). Appena si accende il computer, a tutti gli effetti, ogni comando BASIC Plus e' operativo, non e' necessario quindi dover eseguire la LOAD di questo insieme di routines.

-DOS Plus per la serie 4000 e 8000 che comprende sia il DOS che il BASIC Plus espanso. La sua attivazione e' possibile sempre con il comando SYS(30000).

Vediamo in breve le funzioni del BASIC Plus tenendo presente che nelle formulette tutto cio' che viene scritto fra le parentesi quadre e' optionale e quanto viene richiesto dovra' essere citato solamente quando ne esista la necessita':

AUTO

Provvede alla numerazione progressiva di un programma durante la digitazione.

AUTO [numero di partenza] [,passo]

AUTO la numerazione inizia con 100 con passo 10.
AUTO10 la numerazione inizia con 10, sempre con passo 10.
AUTO10,5 la numerazione inizia con 10 e avra' passo 5.

RENUMBER

Provvede alla rinumerazione di un programma residente in memoria, variando in conseguenza anche le GOTO, le THEN e le GOSUB.

RENUMBER [numero di partenza] [,passo]

RENUMBER senza alcun parametro numerico effettua la rinumerazione del programma residente in quel momento in memoria partendo dalla linea 100 e proseguendo di 10 in 10.
RENUMBER10,10 rinumera il programma oggetto di 10 in 10.

DELETE

Cancella gruppi di linee del programma residente in memoria.

DELETE [numero di partenza] [,numero di arrivo]

DELETE-130 cancella tutte le righe dall'inizio fino alla 130 compresa.
DELETE50-162 cancella tutte le linee fra i numeri interassati compresi.
DELETE355- cancella tutte le linee dalla 355 compresa in poi.

DELETEREM

(solo per serie 4000 e 8000) elimina tutte le voci riportate dopo la parola BASIC REM. Questo comando ci permette di redigere un programma con molte spiegazioni e quindi preparare la versione che a tutti gli effetti dovra' -girare-, quella che dovra' avere una occupazione minore di memoria.
Questo comando non ha optioni limitative, non e' quindi possibile eliminare in parte di programma le REM indesiderate.

FIND

Esegue il display di tutte quelle linee di programma che contengono le lettere che seguono il comando stesso.

```
FIND     elemento BASIC           [,num.part] [,num.arrivo]
FIND "(elemento di stringa)"      [,num.part] [,num.arrivo]
```

FIND A$ esegue la ricerca di tutte le linee di programma che contengono la variabile A$.
FIND A$,50-250 esegue la medesina ricerca dell'esempio precedente, pero' limitando detta ricerca alle linee comprese fra i numeri indicati dopo la virgola. (FIND A$,-100 fino alla riga 100)(FIND A$,500- dalla riga 500 in poi).

APPEND

Esegue l'accodamento di un programma salvato su cassetta o su disco ad un altro residente in memoria, ma non ordinando la numerazione. Per questa ultima ragione risulta obbligatorio eseguire la rinumerazione preventiva del programma che dovra' essere accodato a quello residente in memoria.

```
APPEND ["nom.progr"] [,2]       per cassetta
APPEND "dn:nom.progr",8         per disco
```

DUMP

Tutte le variabili semplici inizializzate vengono edittate nell'ordine in cui sono state create.

DUMPV

(solo per la serie 4000 e 8000) I contenuti di tutti gli Array (matrici (DIM)) vengono visualizzati sul video. Questo comando evita al programmatore di dover eseguire dei cicli di FOR..NEXT, al di fuori del programma, per conoscere i valori contenuti nelle matrici.

HELP

Se durante l'esecuzione del programma questi si "piantasse" sagnalando un SINTAX ERROR od altro messaggio di interruzione anomala con questo semplice comando il sistema visualizzera' la riga di programma che ha causato l'interruzione evidenziando il punto nel quale e' arrivato invertendo da positivo in negativo l'ultimo carattere della singola istruzione.

TRACE

Evidenzia sul lato destro in alto dello schermo il numero delle ultime 6 righe di programma eseguite.

STEP

Comando che abilita automaticamente il TRACE, ma ad ogni riga si ferma ed aspetta che si prema il tasto SHIFT.

OFF

Questo comando disabilita i comandi di TRACE o di STEP.

REPEAT

Attivando questa funzione ogni tasto che viene tenuto premuto per piu' di 1/2 secondo viene ripetuto automaticamente in continuazione fino a che non si rilascia il tasto stesso.
Per l'abilitazione di questa funzione si preme semplicemente il tasto & poi RETURN, mentre per disattivarlo si preme il medesimo tasto shiftato e quindi ancora il tasto di RETURN.
Attenzione che se si carica un qualsiasi programma dalla periferica C2N, il comando REPEAT si disinserisce automaticamente, quindi occorre riabilitarlo per avere ancora la funzione desiderata.
Il comando di REPEAT dovra' essere disabilitato anche se si dovesse eseguire una SAVE su tape-cassetta, in quanto, in caso contrario, questa funzione, inserita, non permette una sicura scrittura del programma che deve essere salvato.

---**H**---

Come caricare in memoria il BASIC Plus

Il BASIC Plus, nella versione su supporto magnetico (nastro o disco) deve, ovviamente, essere caricato prima di diventare operativo; per fare cio' si dovra' eseguire:
LOAD "BASIC PLUS" [,2] se il supporto e' la cassetta magnetica.
LOAD "dn,BASIC PLUS",8 se il supporto e' il disco.
Dopo questa operazione si esegue semplicemente il comando:
SYS (30000)
Ed il sistema rispondera' con una scritta di conferma per convalidare la corretta operazione svolta e la messa in finzioe del BASIC Plus.

---**H**---

nb: tutti i tipi di BASIC Plus sono acquistabili presso tutti i rivenditori Harden autorizzati.

Rendiamo, in oltre, noto a tutti i nostri lettori che il laboratorio specializzato in manutenzione ed installazione, COSMOS 3000 di Paolo Mazzaferro (via Mazzini 26 - 65100 Pescara - 085.31607) ha realizzato due ROM per il PET/CBM serie 3000, contenenti rispettivamente il BASIC Plus e il DOS Support, mentre e' in preparazione la ROM per la serie CBM 8000.
L'onere di acquisto, a nostro parere, e' estremamente irrisorio e potra' essere corrisposto semplicemente tramite controassegno:

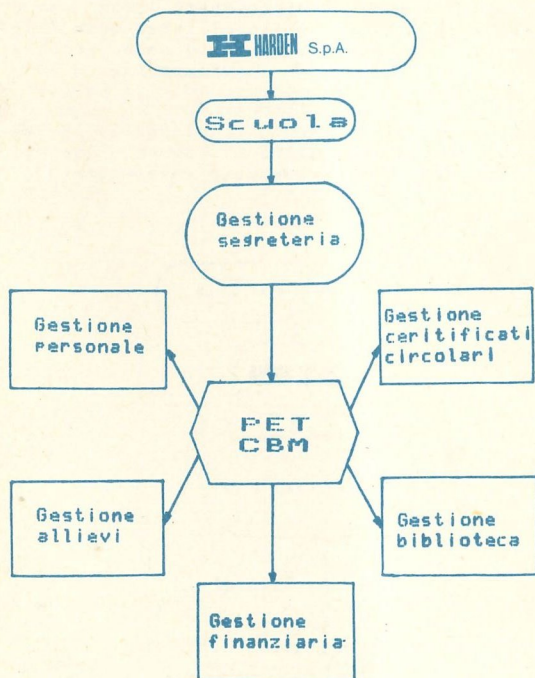
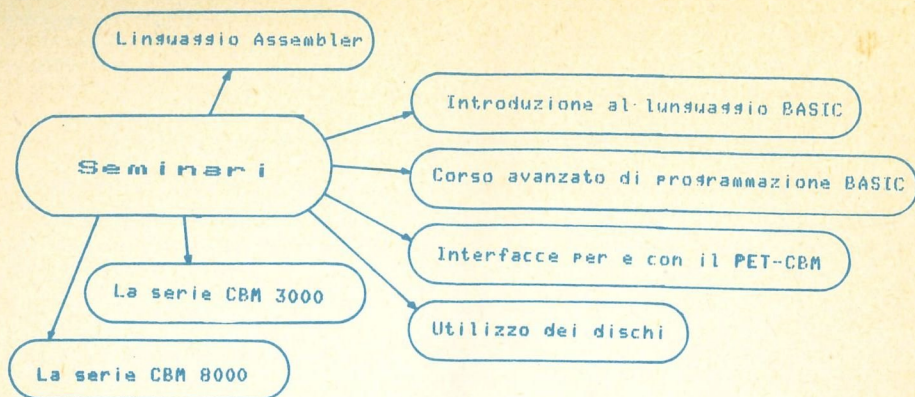
BASIC Plus + DOS Support (2 ROM)	L. 50.000+IVA
Solo DOS Support (1 ROM)	L. 25.000+IVA
Solo BASIC Plus (1 ROM)	L. 35.000+IVA

ROM per serie 8000 in preparazione.

---**H**--- ---**H**--- ---**H**---

Anche il PET va' a scuola

Internamente alla Harden e' stato creato un dipartimento specializzato nel campo educazionale con lo specifico scopo di diffondere e promuovere la cultura dell'informatica nelle scuole.
L'analisi strutturale di questo dipartimento si divide essenzialemente in tre parti ben distinte.

1- Gestione della segreteria, programmi e procedure atte ad aiutare tutto quel lavoro di segreteria che risulta in particolare molto oneroso quale ad esempio le paghe e stipendi.
2- Didattica supportata da moltissimi programmi studiati appositamente da esperti in ogni specifico settore delle materie di insegnamento.
3-Corsi e seminari.

I pacchetti software didattici verranno concessi in uso gratuito alle scuole che utilizzeranno il servizio scolastico di questo dipartimento della Harden.
Questo servizio scolastico e' indirizzato a tutte quelle scuole che accostano gli allievi per la prima volta in modo professionale alla soluzione dei problemi contabili, amministrativi, matematici e fisici attraverso l'informatica.
Gli allievi troveranno facilmente le soluzioni di quei problemi che devono e che dovranno affrontare. L'approccio con l'automazione e l'utilizzo di questi strumenti che oramai gia' ora sono entrati in tutti i settori della vita moderna sara' immediato.
Con le soluzioni proposte dal dipartimento scuole della Harden il problema di immettere gli alunni nella vita quotidiana di lavoro e' senza dubbio risolto.
Un mezzo moderno e valido per un piu' adeguato insegnamento.

Didattica
HARDEN
Scuola
PET CBM

Italiano
Informatica
Costruzioni
Matematica
Lingue straniere
Topografia
Fisica
Higt design
Dattilografia
Tecnica
Chimica
Ragioneria

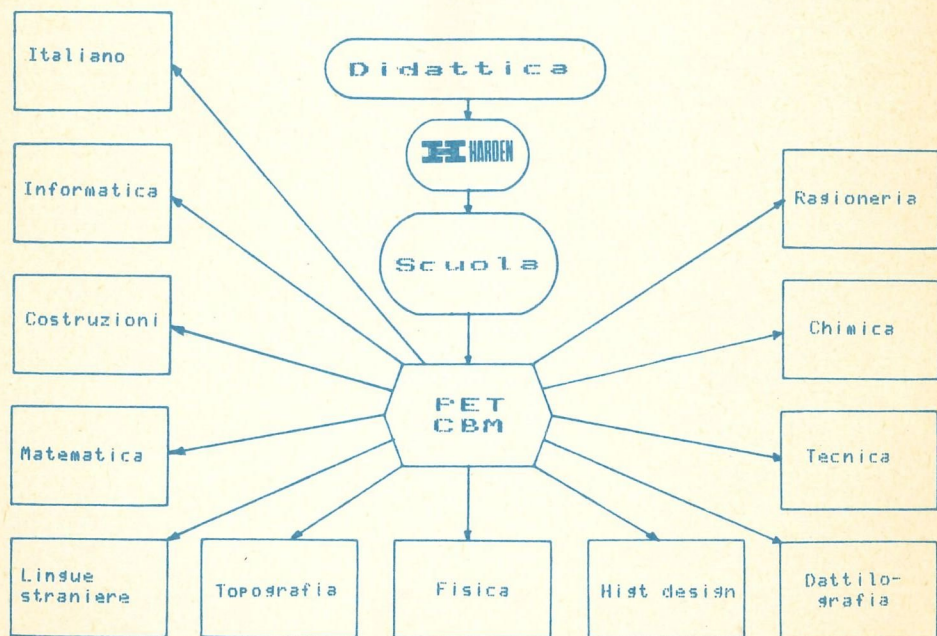



Tavola delle compatibilita'

	2001 old	2001 new R.	2001 32k	3008	4008	4016	3032	4032	8032 32k
Stampante 3022	x	x	x	x	x	x	x	x	x
Stampante 4022	x	x	x	x	x	x	x	x	x
Stampante 8024	x	x	x	x	x	x	x	x	x
Stampante EATON	x	x	x	x	x	x	x	x	x
Stampante LINA 20	x'	x'	x'	x'	x'	x'	x'	x'	x'
Stampante SARA 10	x'	x'	x'	x'	x'	x'	x'	x'	x'
Stampante OLIMPIA	x'	x'	x'	x'	x'	x'	x'	x'	x'
V G 3 2		x	x				x		
Floppy disk 3040		x	x	x	x	x	x	x	x
Floppy disk 4040		y	y	y	x	x	y	x	x
Floppy disk 8050		y	y	y	x	x	y	x	x
Music BOX	x	x	x	x	x	x	x	x	x
Plotter	x	x	x	x	x	x	x	x	x
Tavoletta grafica	x	x	x	x	x	x	x	x	x

x = accoppiamento consentito

y = accoppiamento consentito, ma non sfruttato in pieno

x'= accoppiamento consentito con interfaccia

  = accoppiamento impossibile